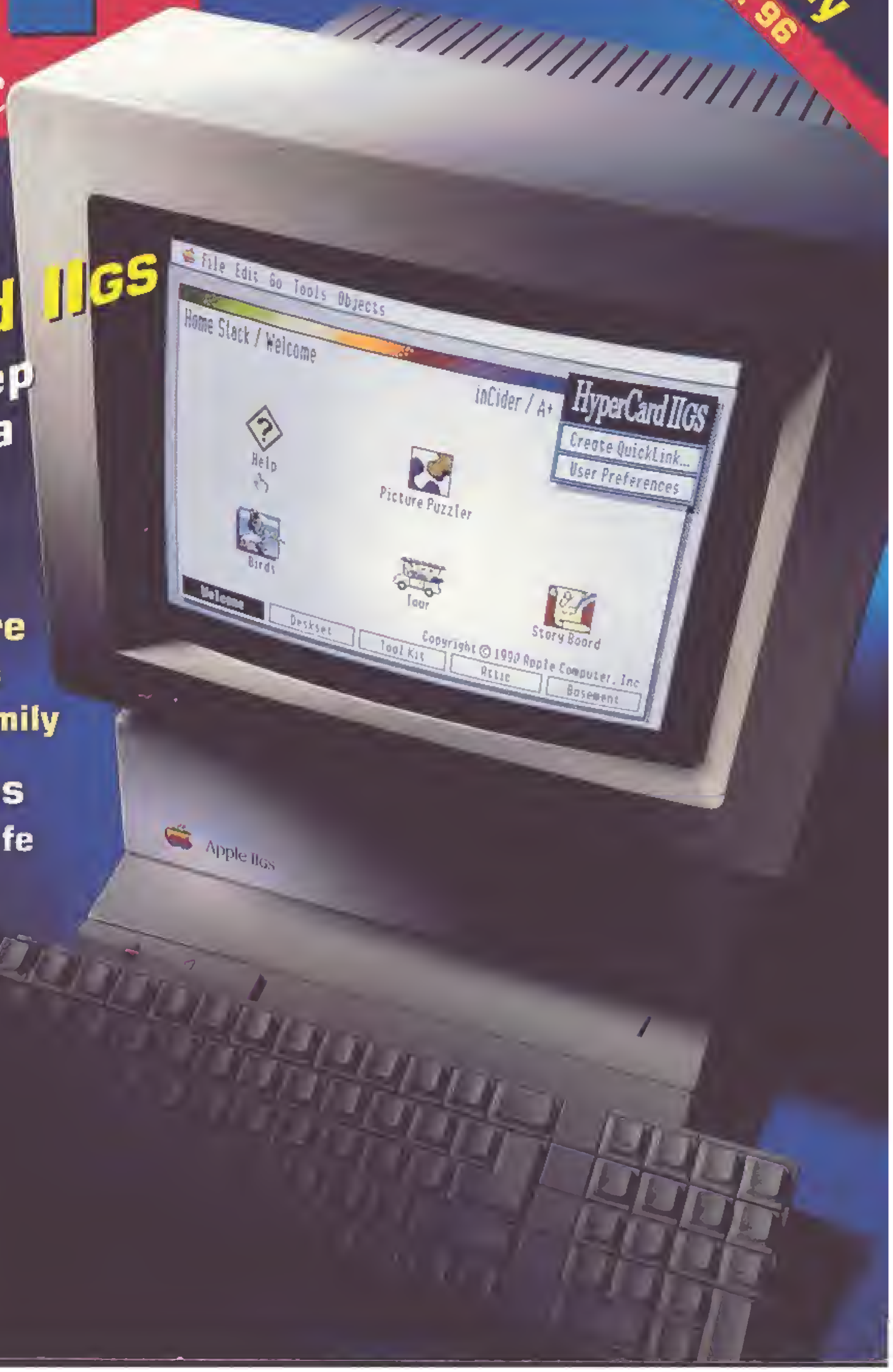
inCider A+ Apple II/Mac

AN IDG COMMUNICATIONS PUBLICATION
February 1991

PIPE DREAMS Go Plumb Crazy
Editors' Choice, p. 96

New! HyperCard IIGS
The Next Step In Multimedia



System Software For GS and Mac
All in the Apple Family

Hard-Disk Drives
The SCSI Side of Life

Reviews:
BannerMania
LogoExpress
Gold Rush!
EuroWorks 3.0

USA $3.95
Canada $4.50

THE APPLE II CULTURE REBORN

By Paul Statt, Senior Editor

HyperCard — Apple's popular multimedia programming product for the Mac — is reviving the do-it-yourself spirit of the Apple II.

The best thing about any Apple computer has always been its accessibility — that anybody with the urge could learn to program and create unique software. But as Apple's computers have become more powerful and more sophisticated, they've also gotten harder to program. The Apple IIGS and the Macintosh, for instance, don't have a native computer programming language built in, as BASIC was in the original Apple II. Like all Apple IIs, the GS comes with Applesoft — but the machine's native mode is the 16-bit 65816.

HyperCard — for the Mac and now for the Apple IIGS — brings the hacker's spirit back to Apple. HyperCard IIGS is a nearly perfect clone of the Mac version — with all the advantages and disadvantages of Apple's Macintosh HyperCard 1.25. It adds color — a feature not found even in HyperCard 2.0, the latest Macintosh version.

HyperCard IIGS, like HyperCard for the Macintosh, is a way to organize information. But it's much more than a database manager. At its most prosaic, HyperCard is a ⇨

Photography • Ed Judice

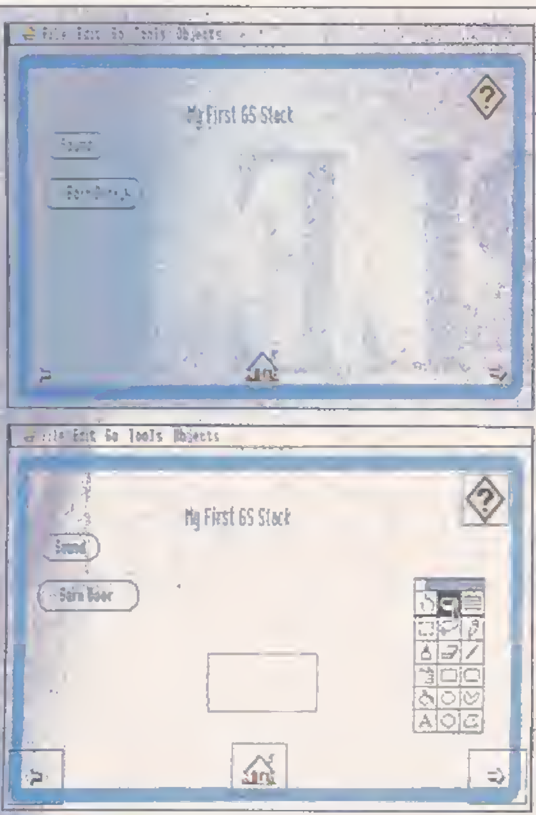
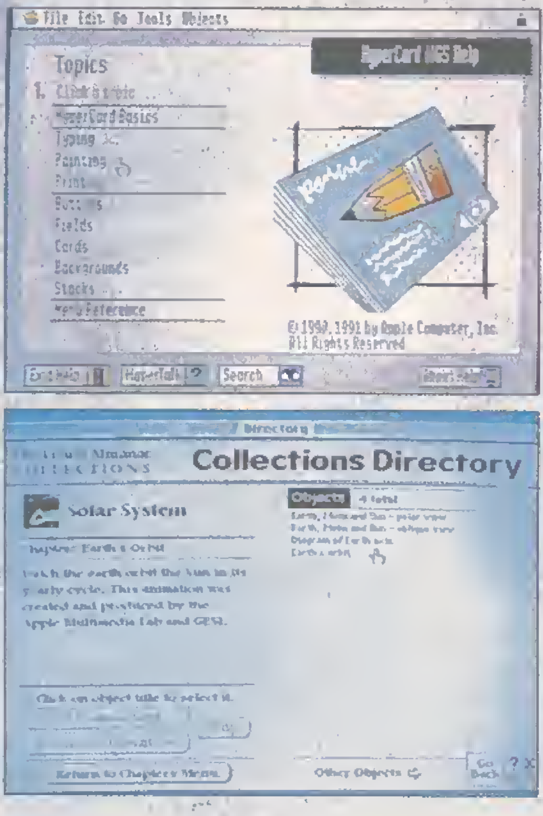


Figure 1.
These two screens both represent the first card of "My First GS Stack." On both cards you see five buttons, clockwise from upper right: "information," "next card," "home," "previous card," "barn door," and "sound." The home, information, next, and previous cards are in the background of this stack: They appear on every card. The text field that reads "My First GS Stack" and the blue border are also in the background. The sound button activates a script that plays a simple tune; the barn-door button initiates a special visual effect. The photo on the bottom shows the same card in "button" mode, which you would use to create or edit a button. Here you also "see" an invisible button, plus the tool menu, which you can tear off from the menu bar.



© 1990, 1991 by Apple Computer, Inc.
All Rights Reserved

Figure 2.
A background, in HyperCard, is emphatically not the scenery. The painting in the HyperCard IIgs Help stack in the top screen isn't in the background; the boxes labeled Exit Help, HyperTalk Search, and About Help are. The background, whether it's made up of buttons, fields, or a whole card, is simply what you want to appear on more than one card. When creating a stack, the background is an essential part of the design. The card on the bottom, part of the stack that controls the Visual Almanac Mac compact disc, is well put together. The lower part of each card — containing all the action items — is the same on each card, while the upper portion is a table of contents for a particular section of the CD. The HyperCard IIgs Help stack is similarly well designed.

programming language. At its most poetic, it's a way of life. As a language it's up-to-date, making use of what's known as *object-oriented programming*.

Programming with an object-oriented language is like building a car on an assembly line: All the parts are there; your job is simply to put them together. The HyperCard lifestyle is as old-fashioned as the original idea of the Apple computer — that anybody should be able to write a program, that businesspeople should write business software, that teachers should write educational software, that kids can even create their own.

As you read about HyperCard IIGS, you may think you have to learn to write HyperCard programs — called *scripts* — to enjoy it. That's not true. Thousands of programmers have already written HyperCard stacks on subjects ranging all the way from ecological awareness to pornography.

Teachers will appreciate HyperCard's five levels of control, from browsing in the stacks (other people's) to scripting your own, with typing, painting, and authoring in between. But just about everybody who uses HyperCard — a program designed to put you in control — will want to write his or her own stacks. Basically, with HyperCard you can do anything the GS can do. Scripting is where the fun begins.

ON THE BUTTON

The key to understanding HyperCard is understanding the role of *cards*, *stacks*, *fields*, *buttons*, and *backgrounds* — the objects the Hyper-Talk programming language gives you to play with.

A card is what you see on a single screen (as in **Figure 1**); a stack is a collection of cards. You could think of it as a stack of index cards, but these cards are live, and you control them. In HyperCard IIGS, a card must fill the entire screen, although it's possible to hide the menu bar at the top. (HyperCard 2.0 on the Mac allows cards of different sizes.) The text and graphics you see on the card appear on two levels: background elements, which carry over from card to card, and the material unique to this card alone.

A background (**Figure 2**) can be as simple as a white screen, or as pretty a picture as you can scan, draw, or paint with HyperCard's built-in graphics tools.

Every card needs a background; in general, as we've noted, you put things you want to see on more than one card into the background. If you don't want to see anything repeated on other cards, you'll include an empty background.

In addition to graphics, most backgrounds contain one or more buttons, the switches or controllers that make something happen (display another card, play music, perform calculations, get help, print, use a modem), and fields, areas containing text or numbers.

Common background buttons include the *home*, *next card*, and *previous card* icons (small pictorial representations, **Figure 3**). One misconception is that a HyperCard stack must have these buttons, but they aren't necessary. (See the accompanying sidebar, " 'Why Did You Do It That Way?' " for a report on button programming.)

Table. Elementary functions used in HyperCard IIGS, as specified by the Standard Apple Numerics Environment on the Apple IIGS' 65816 microprocessor.

log2(x)	computes base 2 logarithm of x
ln(x)	computes natural (base e) logarithm of x
ln1(x)	computes natural logarithm of (x+1)
exp2(x)	computes 2^x
exp(x)	computes e^x
exp1(x)	computes $e^x - 1$
cos(x)	computes cosine of x
sin(x)	computes sine of x
tan(x)	computes tangent of x
atan(x)	computes arctangent of x
random(x)	computes pseudorandom number with x as its seed
compound(r,n)	computes $(1+r)^n$
annuity(r,n)	computes $1-(1+r)^{-n}/r$

Buttons, whether background or unique, can appear as almost any graphic (**Figure 4**). Some don't appear at all — they can be invisible. Some buttons look like the electrical switches on which they're modeled, some look like round rectangles, some look like pictures (*icons*). HyperCard includes a number of icons in a special file you can use in your stacks. An *Icon Editor* stack that lets you draw your own is included with HyperCard IIGS (**Figure 5**). Triad Ventures has also written a desk accessory, included in its **HyperCard IIGS Utilities** package, to help you create icons.

WORD AND IMAGE

Buttons are one way HyperCard gets graphics onto the computer screen, but hypermedia includes good old text, too — in objects called *fields* (**Figure 6**), as we noted above.

If you've had any experience with database ⇨

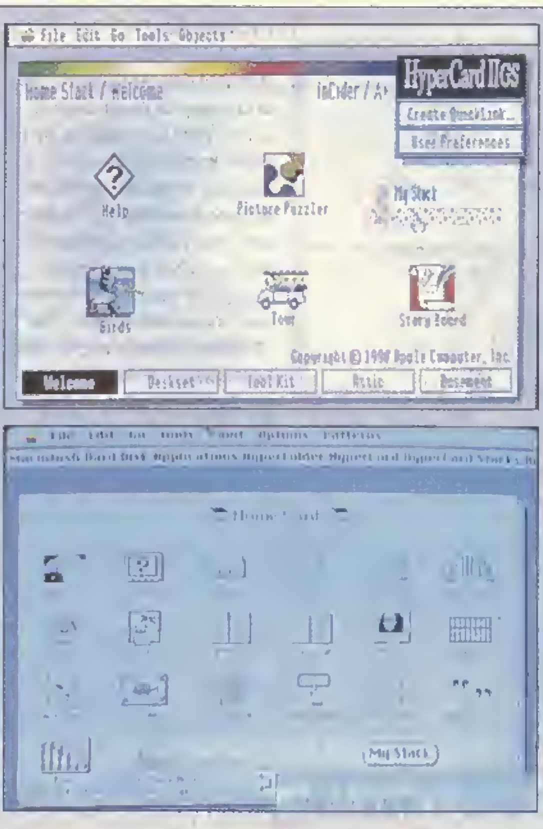


Figure 3.
These are the "home cards," or the first cards you see after starting HyperCard IIGS or HyperCard 1.25 on the Macintosh. A button to access "My Stack" has been added to both home cards. On the Apple IIGS, which uses color, a colorful icon has been created for it; on the monochrome Macintosh, a standard round-rectangle button suffices. The home card for HyperCard 2.0 for the Macintosh looks more like the HyperCard IIGS home card, except that it's limited to black-and-white. On the Macintosh screen (bottom) the user has accessed the "message" (Option-M) feature (also available in HyperCard IIGS) to "set userLevel to 5," which is the highest level possible and lets him or her write HyperTalk scripts.

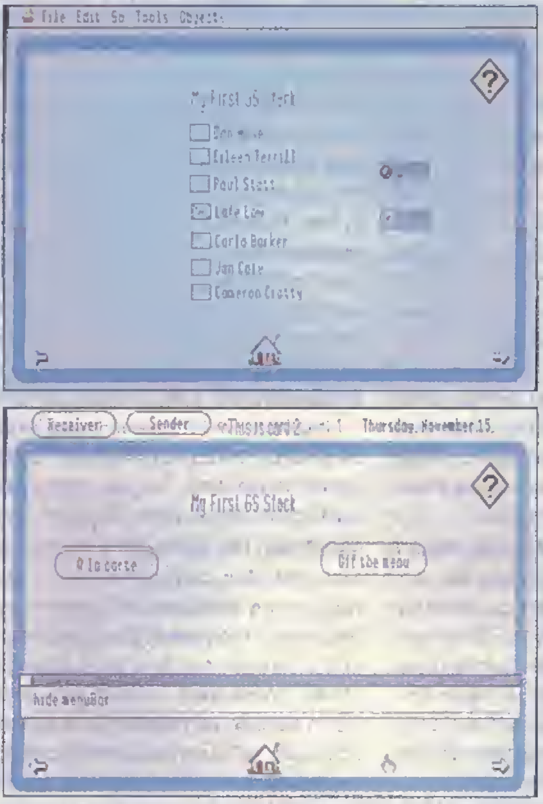


Figure 4.
On these two cards you can see a few of the many kinds of buttons available in HyperCard IIGS. The list of inCider/A+ editors at the top is a "family" of check-box buttons, which means that only one name can be selected at a time. The on and off buttons, which are represented by clip-art icons, are another family. Two buttons on the card on the bottom, "Off the menu" and "A la carte," cause the menu bar at the top of the screen to be hidden and to disappear. ("Hide menuBar" in the message box, Option-M, would have done the same thing.) The text fields with the card's number and the date are "computed" fields that appear on every card — you can see them only on a card whose menu bar is hidden. "Receiver" and "Sender" were practice buttons.

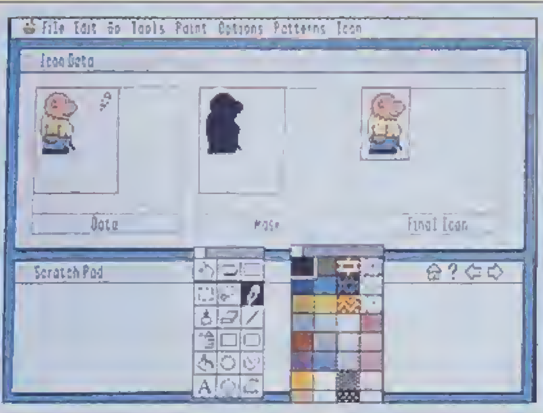


Figure 5.
The Icon Editor program included with HyperCard IIGS is the way to draw icons to use in your own stacks, or in other people's. Whenever you paint or draw in HyperCard, you can tear off the "Tools" and "Patterns" menus from the menu bar and they'll always be available wherever you like on the screen. The "Patterns" menu might have been called "Colors" but for the legacy of monochrome HyperCard for the Mac. The "Home," "Previous," and "Next Card" buttons are in interesting positions in this stack — they don't have to be at the bottom and in the corners.

management, you already know what a field is — a single part of a record. A HyperCard text field, like a database's, can contain either words or numbers; it can appear on a single card, or in every card as a background. You can save the contents of a HyperCard field as ASCII text, and import ASCII text into a stack. HyperCard can also perform calculations, including trigonometric operations, on numbers appearing in a field. (See the accompanying **Table**.)

Stacks are programs written in the HyperTalk language (**Figure 7**). Although HyperTalk doesn't include commands for every possible task — controlling a videodisc player, for example — you can enhance its native abilities with *extended commands*, or *XCMDs*, snatches of code written in C or Pascal that you call from a HyperTalk script. (XCMDs should remind Applesoft BASIC programmers of ampersand commands, which let you call a short machine-language program from BASIC. The effect is similar. See the section below, "Speaking the Language," for more on HyperTalk program-

"Why Did You Do It That Way?"

"In Italy for 30 years under the Borgias they had warfare, terror, murder, and bloodshed, but they produced Michelangelo, Leonardo da Vinci, and the Renaissance. In Switzerland they had brotherly love, they had 500 years of democracy and peace, and what did they produce? The cuckoo clock."
— Orson Welles as Harry Lime, "The Third Man," 1949

While nothing as dramatic as murder and bloodshed occurred during the creation of HyperCard IIGS (well, we did have an earthquake, but that was probably just a coincidence), users are often justifiably curious about the process of designing and creating a program as complex as HyperCard IIGS. Usually the question is posed while screaming for the head of the "idiot who wrote this thing," but nevertheless, what follows are some of the issues we dealt with while writing HyperCard IIGS.

At the most basic level, our major design requirement was to make an Apple IIGS version of HyperCard that was completely compatible with HyperCard 1.25 on the Macintosh computer. At the same time, we wanted to make improvements to the program that would benefit users and take full advantage of the capabilities of the Apple IIGS — hence a color HyperCard with improved printing and reporting capabilities, as well as additional functionality in the HyperTalk scripting language.

A complete description of how we designed and wrote HyperCard IIGS would be extremely long, so as one of the two engineers who worked on HyperTalk, I'm going to confine my discussion to just two features in HyperTalk that I believe represent some of the design decisions we made.

The play command. We agonized over the design of the play command for a long time. In Mac HyperCard, Play lets you utilize the single-voice sound capabilities of that machine. (The play command produces the requested sounds immediately. Multiple play requests are executed sequentially, not simultaneously.) The GS, however, can generate more complex sounds than the Mac.

Naturally, we wanted HyperCard IIGS to be able to take full advantage of the GS' sound capabilities, but we were under several constraints. First and foremost, the GS version had to be completely compatible with the Mac play command. In addition, there were language constraints. HyperTalk, as a language, is supposed to be English-like and simple to read and understand. Whatever syntax we arrived at would have to comprehensible by the average user, not just acoustic engineers. And finally, whatever we came up with had to be useful to the majority of users — we weren't going to add a super-powerful command only three people would need.

Initially we were extremely ambitious, and tried to create a syntax that would give you extensive control over the considerable sound capabilities of the GS. Regrettably, designing a syntax that was powerful, understandable, and backward compatible proved unrealizable, given the time constraints we were under. The language-design issues were difficult because of a rather odd situation — the GS' sound capabilities were too powerful! We simply couldn't figure out a way to unleash all that power and have the syntax understood by nonprogrammers.

We then tried a much simpler approach, a natural extension of the original syntax: one that would let you direct a particular sound to a specified channel, and optionally delay its execution. After setting up several sounds on separate channels, you could then tell the GS to start, and all sounds would play simultaneously. We had high hopes for this approach, but we were tripped up by the problem of synchronizing multiple sounds. Specifically, we had trouble resolving the language issue (and it's a lot more involved than you might think): how to accurately make sound B start 10 seconds, or even harder, 10 beats, into sound A.

Deadlines were looming, so rather than

ming.) You can attach an XCMD either to a single stack or to the HyperCard program itself. Triad Ventures, for instance, has already written some XCMDs to facilitate playing music (available in its **MIDI Music** package).

PLUS AND MINUS

One of HyperCard's secret strengths is that it's a pretty good database manager (**Figure 8**). Apple plays that down because it doesn't want software companies to complain that Apple's controlling the market.

Besides, "information engine" sounds more glamorous than "database manager." But teachers who need class records and individuals who don't need relational structured query language (SQL) just to keep Christmas card lists will find that they can write or buy HyperCard stacks that do the job.

Actually, HyperCard IIGS' number-crunching power would make some database managers envious. Financial functions such as annuity calculation and compounding for computing interest, and mathematical functions such as natural logarithms and trigonometric functions (see the accompanying **Table**), are available — and you won't find those in AppleWorks Classic. HyperCard IIGS uses *Standard Apple Numerics Environment* (SANE) protocol, which gives you access to a wealth of functions, and can speed up calculations considerably if you add a math coprocessor to your GS.

The program's most obvious strength is that your GS will be able to run stacks created for the Mac, and your Mac will be able to run GS stacks. At press time, the pair of stacks that perform the conversion, HyperMover GS and HyperMover Mac, weren't available. But if you compare the look of a Mac stack and its Apple IIGS cousin, you can see that converting one to the other shouldn't be too difficult.

We tried to copy the text of a HyperCard program, called a *script*, from a GS to a Mac LC via an AppleTalk network, but it wasn't quite that simple. The text arrived intact, but the program didn't quite work on the Mac. You'd need to ⇨

"Mac or GS, HyperCard brings the hacker's spirit back to Apple."

do a mediocre job on a new syntax, and thus a disservice to the users and to the capabilities of the machine, we decided to stay with the original single-voice syntax from the Mac.

Not wanting to neglect all that power, however, we added two new XCMD callbacks: BeginXSound and EndXSound. XCMDs are user-created extensions to HyperTalk; callbacks are routines that let HyperCard and XCMDs communicate.

BeginXSound lets an XCMD take over the GS sound chip completely; EndXSound gives control back to HyperCard. With these new callbacks, you can extend the sound capabilities of HyperCard IIGS as far as you want, and without the constraints we were under.

You can write sound XCMDs that are as specialized, powerful, or weird as you like, and we strongly urge you to do so. That's what XCMDs are for.

Button families. The members of the HyperCard IIGS engineering team are also experienced Mac HyperCard users, and thus had a great deal of experience creating stacks with sets of radio buttons. Radio buttons, as defined by Apple's Human Interface Guidelines, are used to select one, and only one, choice out of many. If one radio button is highlighted, the other radio buttons must by definition be dark. And each button has to have a script that handles this procedure.

Imagine a stack with five radio buttons. To act properly, button 1's script would look like the following:

```
on mouseUp
  set the hilite of button 1 to true
  set the hilite of button 2 to false
  set the hilite of button 3 to false
  set the hilite of button 4 to false
  set the hilite of button 5 to false
end mouseUp
```

Button 2's script would look something like this:

```
on mouseUp
  set the hilite of button 1 to false
  set the hilite of button 2 to true
  set the hilite of button 3 to false
  set the hilite of button 4 to false
  set the hilite of button 5 to false
end mouseUp
```

And so on. Not very difficult, certainly, but a terrible nuisance. As longtime HyperCard users, we realized that a lot of tedious scripting could be avoided if the machine could handle highlighting and dimming automatically.

As it happens, the Apple IIGS Toolbox made this relatively easy to implement, and thus button families were born. By utilizing button families, a stack developer doesn't have to write any code at all to handle the proper highlighting of radio buttons.

An interesting thing happens when an experienced Mac HyperCard user sees HyperCard IIGS. While the obvious features, such as color, impress them, what really excites them are button families, and for the same reason they excite us — they save HyperTalk scripters lots of time.

As it happens, we decided to make families a property of all button types, not just radio buttons. That's of course their most obvious use, and although there's a danger that by not restricting their use, button families may be used for evil purposes (that is, in violation of Apple's Human Interface Guidelines), we felt it was best to give program writers the extra flexibility. In fact, if you look at their scripts, you'll notice that the rectangular buttons along the bottom of the home stack are implemented with button families.

A personal note. We brought a lot of personal experience as HyperCard users to the table when determining which features should be added or improved for the GS version. This program was written by HyperCard users. We're pretty proud of HyperCard IIGS, and hope you enjoy it and get a lot of use out of it.

— John Lawler, Apple Computer

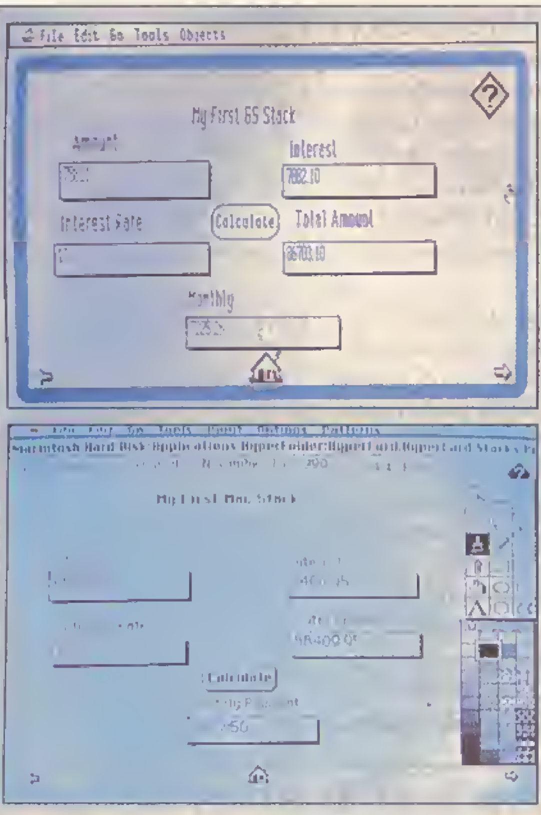


Figure 6.
At the top, HyperCard IIGS; on the bottom, HyperCard 1.25 for the Mac. This card uses the mathematical muscle of HyperCard to compute, in a not very elegant way, the monthly payments on a year's loan. The HyperTalk script for the "Calculate" button is identical on both cards — it was copied from Apple's HyperTalk Beginner's Guide. This demonstrates the relative ease with which Mac and Apple IIGS stacks can be converted to each other's format. With the exception of the background buttons and the "Calculate" button, all objects on these cards are fields.

HyperCard IIGS script

```
on mouseUp
   set numberFormat to 0.00
   get card field "Amount"
   multiply it by card field "Rate"
   divide it by 100
   put it into card field "Interest"
   add card field "Amount" to it
   put it into card field "Total"
   divide it by 12
   put it into card field "Monthly"
end mouseUp
```

HyperCard 1.25 script

```
on mouseUp
   set numberFormat to 0.00
   get card field "Amount"
   multiply it by card field "Rate"
   divide it by 100
   put it into card field "Interest"
   add card field "Amount" to it
   put it into card field "Total"
   divide it by 12
   put it into card field "monthly"
end mouseUp
```

Figure 7.
HyperTalk scripts for the HyperCard IIGS and HyperCard 1.25 versions of a button that calculates interest (Figure 6) are identical; except that in the Mac script, the lines "on mouseUp" and "end mouseUp" appear automatically; in HyperCard IIGS you need to write them out. HyperTalk sounds enough like English that reading a script is painless; writing in HyperTalk requires more discipline.

change some commands, but you could do the translation in a mechanical way — doing a find-and-replace on the offending structures. Tedious for you, but a snap for HyperMover.

The HyperMover programs will be available to developers and user groups. But other challenges remain: Stack conversions will pose a tricky legal question, for one thing. If you buy, say, a company's GradeBookStack for Macintosh HyperCard, do you need to buy another copy or a site license to run it on HyperCard IIGS? If you can legally use any Mac stack, a lot of stacks — commercial as well as public-domain — will be available for HyperCard IIGS.

A number of stacks are already available for Roger Wagner's **HyperStudio** — you don't even need to own HyperStudio to use some of them if they were created with the runtime version of the program. (See "Shareware Solutions," What's New, p. 20 in this issue.) That won't be possible with HyperCard.

It remains to be seen how many stacks will be written for HyperCard IIGS. Mac HyperCard ignited interest initially among commercial developers, but that burned out rather quickly. At press time, we knew the Boston Computer Society was writing a disk of GS shareware stacks, and Triad Ventures had written a commercial **MIDI Music** stack using Apple's new GS **MIDI Synth** tool.

AUDIO AND VIDEO

Given the musical talent of the GS, it's a shame HyperCard IIGS can't sing on its own, or at least play the piano. The version we tested had a choice of two musical instruments: harpsichord and a "boing" sound, which simply doesn't sound very musical. A Scripter's Tools stack included with HyperCard IIGS, however, contains 11 other sound resources for instruments such as piano and guitar, which you can attach to a single script or to the program itself. (See the sidebar, " 'Why Did You Do It That Way?' " for details.)

HyperCard doesn't have built-in sound digitization, however, as HyperStudio does. (See **Figure 9**. HyperStudio also includes a microphone, much like the Mac LC, making it easy to add sound to a stack.) Is Apple avoiding putting too much musical muscle into any of its products because it fears legal complications with Apple Corps, the Beatles' record company?

As we noted above, the current version of HyperCard IIGS also doesn't include commands for running a videodisc player, as HyperStudio does. HyperStudio has to offer lots of extras, because, lacking a HyperTalk-style programming language as it does, adding features to the program is hard.

SPEAKING THE LANGUAGE

HyperCard is a general-purpose database manager, paint program, programming language, and even something of a word processor. The drawback? HyperCard — either version — is slow. HyperTalk simply adds another layer between you and the machine. HyperTalk is an interpreted language, which means that every line of code must be translated into something the GS understands before the machine starts to do anything. The advantage of an interpreted language is that the same code works on different machines. That's especially clear in the case of HyperCard: It's easy to write code that works on both the GS and the Mac.

HyperTalk is a plain-speaking, unaffected programming language, but a programming language nevertheless. You have to mind your Ps and Qs when you write a HyperCard stack — make sure you spell everything correctly and leave the right number of spaces.

HyperTalk is forgiving in some ways. It's unusual in that it lets you use two different words in certain cases to mean the same thing — *slow* and *slowly*, or *go to* and *go*, for instance. But it's about as much like spoken English as the commands players type into a text-adventure game. It requires some discipline to make yourself understood.

Pricing and Availability

At press time, final decisions about the price and distribution of HyperCard IIGS hadn't been made. Jane Lee, product manager for HyperCard IIGS, told us in November, though, that the price "should be around $99." That's what HyperCard for the Macintosh costs. "It will initially be distributed by Apple," Lee said, in contrast to HyperCard Mac, which is now distributed, in part at least, by Claris, Apple's software subsidiary. "It will be available through dealers and APDA [the Apple Programmers and Developers Association, phone (408) 562-3910, (800) 282-3732], and we are pursuing licensing by [Apple-authorized] user groups."

Lee noted that because "you can't use [HyperCard IIGS] out of the box with your new [Apple IIGS] CPU — it needs 1.5 megs and a hard drive or network to run — HyperCard IIGS will not ship with the CPU," as Macintosh HyperCard does.

In addition, Lee said, "HyperMover will be available to developers and through APDA."

— P.S.

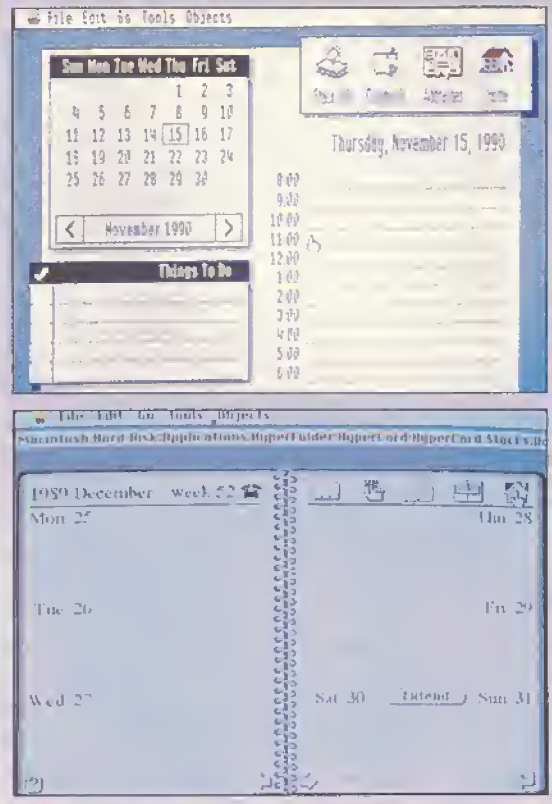
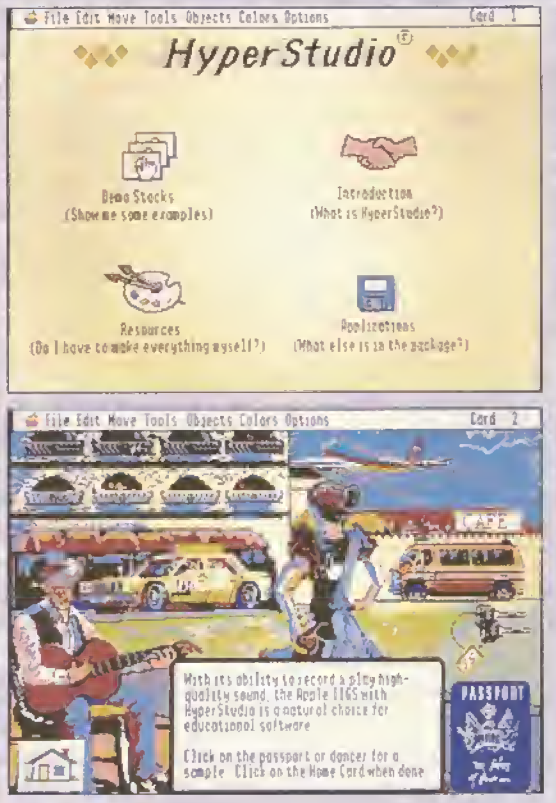


Figure 8. Calendar stacks for HyperCard IIGS (top) and Mac HyperCard 1.25 (bottom). HyperCard handles a small job like your personal appointment schedule easily; a more taxing task might overpower it. HyperCard IIGS includes this calendar stack and one to file addresses and phone numbers.



With its ability to record & play high-quality sound, the Apple IIGS with HyperStudio is a natural choice for educational software.

Click on the passport or dancer for a sample. Click on the Home Card when done

Figure 9. HyperStudio for the Apple IIGS is a program much like HyperCard IIGS, which has been available for two years already. Its home card, top, is quite a bit like HyperCard's, but stacks aren't compatible. On the bottom, a sample card from a HyperStudio stack. On this card, designed to teach the Spanish language, you can click on the dancer to hear a prerecorded human voice say "la senorita." This kind of sound recording isn't available in HyperCard IIGS.

"At its most poetic, HyperCard is a way of life."

HyperStudio is easier to use than HyperCard, if by that you mean you can write a program without writing a line of code. HyperStudio never gets to the scripting level. It's indeed possible to build a HyperCard IIGS stack without HyperTalk by simply combining elements, but it's like baking a cake from a mix — the result will be edible, but bland. In any case, HyperTalk isn't exactly difficult. It strives to look like English, and although the impersonation isn't always successful, the meaning of a line such as *put the date into field "Today"* should be pretty clear. You'll need to take some time, and perhaps read a book or two, before you can write a script in HyperCard.

Addison-Wesley publishes excellent HyperCard references, and at press time also had a HyperCard IIGS book in the works. You may need to spend a weekend learning about it; an evening might do it. But you won't need to take a college-level course as you would to learn Pascal or C.

If you don't want to type even one line of program code, HyperStudio can do much of what HyperCard can, and some things it can't (such as videodisc control and sound digitization), simply by pointing and clicking a mouse. Some teachers will want their students to have the chance to create stacks without programming; others may want students to learn to write a computer program.

An interesting aspect of HyperCard, GS or Mac, is that you never need to save a stack. If you've experienced a crash or two, you'll understand the wisdom of this feature in a programming language. HyperStudio, which lacks the language, lets you decide when and if you want to save. If you want to experiment with HyperCard, you have the option to *Save a Copy* of your work up to a certain point.

A BIG PROGRAM

To use HyperCard, you need a big GS: 1.5 megabytes of memory and a hard-disk drive or network. Those are the official system requirements. We used a 1.25-megabyte GS with a 40-megabyte hard drive to test a beta version of the program. It was usable with that setup, but delays and out-of-memory messages were frustrating. The system ran more smoothly with 4 megabytes of RAM and an accelerator.

Faster speed really isn't crucial, but the hard drive is. After all, the whole point of HyperCard is that it's a way to manage lots of information — it gets interesting only when you have lots of cards. Some of the best Mac stacks, for instance — such as **The Visual Almanac**, shown in **Figure 2** — run from CD-ROM discs with half a million megabytes. An AppleTalk network would be a good way for a school to run HyperCard IIGS.

Not everyone has hard-disk drives, CD-ROM players, and local-area networks, though. That's the crucial advantage HyperStudio has over HyperCard IIGS: HyperStudio runs on a GS with 512K of RAM and a 3.5-inch floppy-disk drive. You can even create stacks that run with less memory, and run without HyperStudio.

But let's not take anything away from the supreme achievement of HyperCard IIGS — it's a Macintosh program that runs on the Apple IIGS. The hundreds of Mac HyperCard programs that are available are now also available for the GS. HyperCard IIGS should actually help make HyperCard legitimate, and help maintain Apple's dominance in the school computer market.

The irony is that HyperCard may eventually help make the Macintosh legitimate in schools. Schools that have GSes can now use HyperCard IIGS to run existing Mac stacks, many of which are designed for education. Schools, or teachers, who buy Macs can use the new machines to write educational software that will run on both Macs and Apple IIGSes. The twain have met. ❑

PRODUCT INFORMATION

Apple II Public-Domain Disks: HyperCard IIgs Stacks
Apple II SIG
Boston Computer Society
One Center Plaza
Boston, MA 02108
(617) 227-4636 x201

HyperCard Stack Design Guidelines, $16.95
HyperCard Script Language Reference, $22.95
Addison-Wesley Publishing
Route 128
Reading, MA 01867
(617) 944-3700

HyperCard IIgs
Apple Computer
20525 Mariani Avenue
Cupertino, CA 95014
(408) 974-1010
$99

HyperCard IIgs Utilities
HyperStuff Clip Art
MIDI Music
Triad Ventures
PO Box 349
Smithtown, NY 11787
(516) 732-3771
(516) 360-0797
$39.95 each

HyperStudio 2.1
Roger Wagner Publishing
1050 Pioneer Way
Suite P
El Cajon, CA 92020
(619) 442-0524
$149.95
$10 demo disks

HyperStudio Network
P.O. Box 103
Blawenburg, NJ 08504
(609) 466-3196
$29/year

MIDI Synth
Apple Programmers and Developers Association
Apple Computer
20525 Mariani Avenue
MS-33G
Cupertino, CA 95014
(408) 562-3910
(800) 282-3732
$35

The Visual Almanac
Optical Data
30 Technology Drive
Warren, NJ 07059
(908) 668-0022
$100